QoS : tc

INTRODUCTION

Les flux amenés à transiter sur les réseaux sont extrêmement variés en type, en volume et en importance. Lorsque le réseau est encombré, si aucune hiérarchie entre les différentes informations n'est définie, les flux sont impactés de manière aléatoire. Dans ce cas, il peut être utile de mettre en place une politique de QoS (*qualité de service* dans la langue de Molière) qui permettra de s'assurer que certains flux seront rendus prioritaires par rapport à d'autres.

De nombreuses technologies de gestion de la qualité de service ont été développées pour les réseaux IP, comme IntServ et DiffServ, qui servent tous deux à simuler une commutation de cellules sur un chemin garantissant la qualité de service. Cependant, ces technologies sont souvent utilisables uniquement au sein des réseaux professionnels (au niveau des FAI, ou pour les réseaux d'entreprises de grande taille) et il est rare que ces technologies soient accessibles de bout en bout sur un réseau comme Internet.

Le but du TP, est de mettre en oeuvre une politique de gestion de la qualité de service au niveau des couches réseau d'un système d'exploitation GNU/Linux, afin de pouvoir garantir la priorité de certains flux par rapport à d'autres.

PRINCIPES

La gestion QoS de Linux et le contrôle du trafic (IProute) permettent de limiter la masse de données circulant sur une interface réseau. Cette limitation sur un certain type de connexion réseau aura pour effet de libérer des ressources pour d'autres types. En principe, le contrôle du trafic réseau sortant d'une interface est plus aisé que le contrôle du trafic réseau entrant. Il est important de relever ce détail, car sur une machine faisant office de passerelle et possédant deux interfaces, il sera plus simple de contrôler les sorties de l'une que les entrées de l'autre.

Le contrôle du trafic sous Linux repose sur un principe de base selon lequel toutes les données arrivent via un "gros tuyau". Ces données peuvent être dispatchées dans plusieurs canaux (tubes, files d'attente, ou tout autre symbolique qui convient) en fonction d'une sélection préalable. Il est possible d'effectuer cette classification par protocole, port, provenance, destination, etc.

Pour atteindre cet objectif, il faut tout d'abord expliquer la façon dont la qualité de service est gérée sous Linux.

Lorsqu'un paquet IP est envoyé depuis une application, il transite par la table MANGLE d'iptables. À ce moment, il est possible d'effectuer des modifications sur le paquet IP, comme une réécriture d'adresse, la modification d'un champ IPv4 ou le marquage de celui-ci. Pour la gestion de la QoS on va marquer les paquets, soit en modifiant la valeur du champ TOS du paquet IPv4, soit en le marquant virtuellement avec MARK. Dans notre cas, nous effectuerons un marquage virtuel en fonction du port de destination.

Lorsque les modules de QoS sont chargés dans le noyau Linux, le paquet est ensuite traité par le module Gestion de la qualité de Service IP sous Linux. Ce module permet alors de gérer les paquets dans des séries de file d'attente personnalisables, en fonction de la valeur de leur marque.

PRÉ-REQUIS

La gestion de QoS sous GNU/Linux demande un support particulier dans la configuration du kernel des modules de QoS.

NetFilter

NetFilter est intégré à Linux depuis sa version 2.4.

```
# uname -a
Linux pccop0b004-14 2.6.24-umlv #1 SMP Tue Jun 3 12:49:33 UTC 2008
i686 GNU/Linux
```

Chargement des modules

Pour activer la gestion de la qualité de service (QoS – Quality Of Service) sous Linux, il faut que les options CBQ, SFQ et HTB aient été compilées comme modules du noyau.

```
# modprobe -l | grep cbq
/lib/modules/2.6.24-umlv/kernel/net/sched/sch_cbq.ko
# modprobe -l | grep sfq
/lib/modules/2.6.24-umlv/kernel/net/sched/sch_sfq.ko
# modprobe -l | grep htb
/lib/modules/2.6.24-umlv/kernel/net/sched/sch_htb.ko
```

Il faut également les charger à l'aide des commandes suivantes :

```
# modprobe sch_cbq
# modprobe sch_sfq
# modprobe sch_htb
```

IPRoute

Pour pouvoir utiliser l'utilitaire TC, qui permet de gérer les files d'attente au niveau du noyau Linux, il faut installer le paquetage Debian iproute :

```
# aptitude install iproute
```

IPRoute est composé principalement des utilitaires TC et IP.

IP permet d'effectuer des configurations complexes de routage, il est beaucoup plus puissant que l'utilitaire ROUTE qui permet d'appliquer des règles de routage basiques.

TC permet de contrôler les fonctions de QoS intégrées au noyau. Il peut sélectionner des types de paquets IP et les envoyer vers différents types de files. Il permet aussi de sélectionner les paquets en fonction de leur marquage *Netfilter*. Cette solution est de loin la meilleure car elle offre à la QoS toute la puissance de sélection de paquets de *Netfilter*.

MGEN

MGEN est un outil de génération de trafic réseau simple d'utilisation. Il permet de générer des émissions constantes de paquets UDP et TCP (mode CBR), mais également de générer des trafics aléatoires variables selon des lois probabilistes, comme les lois exponentielles et les lois de Poisson. Il permet également de créer rapidement un écouteur permettant de recevoir ces trafics.

Cet outil est déjà présent sur les machines utilisées.

Toutefois, dans le cas où nous aurions dû l'installer, il faut savoir qu'il n'est pas disponible sous forme packagée pour Debian GNU/Linux. Il est donc recommandé de le télécharger et le compiler soi-même.
La dernière version (bêta) disponible à la date d'écriture de ce rapport est la 4.2 bêta 6.
On peut l'installer comme suit :

```
# wget http://downloads.pf.itd.nrl.navy.mil/mgen/src-mgen-4.2b6.tgz
# tar xzf src-mgen-4.2b6.tgz
# cd mgen-4.2b6/unix
# make -f Makefile.linux mgen
```

L'exécutable **mgen** est alors stocké directement dans le dossier mgen-4.2b6/unix. Pour qu'il soit disponible dans votre PATH, il faut soit le copier dans le dossier /usr/local/bin, soit ajouter ce répertoire au PATH.

> /!\ Lors de la création du fichier d'auto-configuration pour MGEN, notamment sous Windows, il est primordial de laisser une ligne vide à la fin du fichier pour que la dernière commande soit exécutée.

Gnuplot

Cet outil permet de dessiner rapidement des courbes à partir de séries de chiffres. Il est extrêmement simple d'utilisation et permet d'interpréter rapidement des résultats parfois complexes. Ce programme peut s'installer à l'aide de la commande :

```
# aptitude install gnuplot
```

TRPR

TRPR est un outil qui permet de traiter les données recueillies par MGEN afin de les transformer, entre autres choses, dans des formats compatibles avec GNUPLOT. Il permet également de générer des statistiques intéressantes comme des moyennes de gigue, des moyennes de débit ou des moyennes sur le taux de perte de paquets UDP.
Aucune version précompilée n'est disponible, et son code source tient en un seul fichier C++. Pour le compiler, il suffit de faire comme suit (pour la version 2.0 bêta 2) :

```
# wget http://downloads.pf.itd.nrl.navy.mil/proteantools/src-trpr-2.0b2.tgz
# tar xzf src-trpr-2.0b2.tgz
# cd TRPR
# g++ -o trpr trpr.cpp –lm
```

Le switch `-lm` permet de lancer également le *linker*, de sorte que l'objet compilé soit linké et mis au format standard Linux ELF.

De même que pour MGEN, il faut soit copier **trpr** dans votre répertoire /usr/local/bin, soit placer le répertoire TRPR dans le PATH.

MISE EN PLACE D'UN ARBRE DE CONTRÔLE

Le but de la manipulation est d'implémenter des restrictions de flux et de partage en utilisant la commande TC. Les tests seront fait entre deux machines en utilisant MGEN.

Nous allons mettre en place 4 files d'attente configurées selon les modalités suivantes :

- pour la marque 1, une file de discipline SFQ avec un trafic limité à 10 kb/s et un trafic normal à 10 kb/s ;
- pour la marque 2, une file de discipline SFQ avec un trafic limité à 1 Mb/s et un trafic normal à 200 kb/s ;
- pour le TOS 3, une file de discipline SFQ avec un trafic limité à 1 Mb/s et un trafic normal à 700 kb/s ;
- l'ensemble de ces files est limité à 1 Mb/s au total, toutes confondues ,et la discipline de cette file de sortie est HTB2.

Par conséquent, le trafic sortant de la machine sera limité, sur les ports 5000, 5001 et 5002 à 1 Mb/s au total, et le traitement par files d'attentes séparées permettra de répartir le trafic selon son importance.

Nous allons donc mettre en place l'arbre de contrôle du flux suivant :

- qdisc HTB : Racine **(1)**
  - class HTB : limitation de trafic à 1Mb/s **(2)**
    - class HTB : Limitation de trafic à 10kb/s avec un trafic normal à 10kb/s **(3)**
      - qdisc SFQ : partage équitable de la bande passante entre les connexions. **(6)**
      - filter : le port destination 5000 **(9)**
    - class HTB : Limitation de trafic à 1Mb/s avec un trafic normal à 200kb/s **(4)**
      - qdisc SFQ : partage équitable de la bande passante entre les connexions. **(7)**
      - filter : le port destination 5001 **(10)**
    - class HTB : Limitation de trafic à 1Mb/s avec un trafic normal à 700kb/s **(5)**
      - qdisc SFQ : partage équitable de la bande passante entre les connexions. **(8)**
      - filter : le port destination 5002 **(11)**

Script de mise en oeuvre automatique

Voici les commandes nécessaires, en prenant en compte :

- 10 kb = 10240 b
- 200 kb = 204800 b
- 700 kb = 716800 b
- 1Mb = 1048576 b

```
# Définition des classes de Qualité de Service
# tc qdisc replace dev eth1 root handle 10: htb
tc qdisc del dev eth1 root
tc qdisc add dev eth1 root handle 10: htb (1)
tc class add dev eth1 parent 10: classid 10:1 htb rate 1048576 (2)
tc class add dev eth1 parent 10:1 classid 10:10 htb rate 10240 ceil 10240 (3)
tc class add dev eth1 parent 10:1 classid 10:20 htb rate 204800 ceil 1048576 (4)
tc class add dev eth1 parent 10:1 classid 10:30 htb rate 716800 ceil 1048576 (5)

# Activation du partage équitable pour chaque file
tc qdisc add dev eth1 parent 10:10 handle 40: sfq (6)
tc qdisc add dev eth1 parent 10:20 handle 50: sfq (7)
tc qdisc add dev eth1 parent 10:30 handle 60: sfq (8)

# Définition du port destination
tc filter add dev eth1parent 10: protocol ip u32 match ip dport 5000 0xffff flowid 10:10 (9)
tc filter add dev eth1parent 10: protocol ip u32 match ip dport 5001 0xffff flowid 10:20 (10)
tc filter add dev eth1parent 10: protocol ip u32 match ip dport 5002 0xffff flowid 10:30 (11)
```

`qdisc` signifie que nous sommes en mode *Queue Discipline* : c'est un algorithme gérant une file d'attente. L'objet `qdisc` contient le type de l'algorithme et la file d'attente.

`filter` est utilisé par un `qdisc` contenant des classes afin de déterminer dans quelle classe doit être empilé le paquet. Chaque fois qu'un trafic arrive dans une classe possédant des sous-classes, cela nécessite une classification. Il est important de noter que les filtres résident dans les `qdisc`. Les filtres ne sont pas maître de ce qui arrive.

Le mot clé `add` signifie qu'un `qdisc`, une `class` ou un `filter` est ajouté à un noeud. Pour ce qui est des `qdisc`, le mot clé `del` sert à supprimer l'entrée répondant aux caractéristiques qui suivent.

Les `qdisc` peuvent être attachés à la racine (`root`) d'une interface (`dev`) que ce soit avec `del` ou `add`.

Un `qdisc` qui peut avoir des fils peut leur affecter un identifiant appelé `handle`.

L'identifiant qui suit le mot clé `handle` est l'identifiant qui sera affecté aux fils. Dans notre cas il s'agit de `10` qui est la valeur en général choisi lorsque le `qdisc` est attaché à la racine.

Quelque soit le type (`qdisc`, `class` ou `filter`) un `parent` peut être précisé en passant son identifiant ou en attachant directement la racine d'une interface, comme précisé précédemment. Lors de la création d'un `qdisc` ou d'un `filter`, il peut être nommé avec le paramètre `handle`. S'il s'agit d'un `qdisc` avec classes, il sera nommé avec `classid`. Les `parent` sont numérotés avec l'identifiant qui suit le mot clé, on précise ensuite le type de nommage (classid ou handle). L'identifiant est précisé avec celui saisi après le mot clé.

Il est possible de préciser la bande passante et le débit à l'aide des paramètres `rate` et `ceil`. L'unité de mesure est précisée en suivant la valeur (exemple : 10kbps, 1Mbps, …).

Après le mot clé `protocol` suit le type de protocole qui est employé, dans notre cas il s'agit de l'Internet Protocol.

Enfin `flowid` permet d'affecter un identifiant au flux. Cet identifiant correspond à un des `parent` qui ont été précisés dans les commandes précédentes.

Pour afficher les opérations effectuées par tc sur l'interface *eth1*, on passe la commande :

```
tc -s -d class show dev eth0
```

Cela signifie que les opérations effectuées sur l'interface *eth1* vont être affichées.

Marquage des paquets

Pour marquer les paquets des différents flux, on ne va pas utiliser champ TOS des paquets IP mais un marquage virtuel avec MARK.

C'est un champ numérique, en lecture/écriture, attaché au paquet par le noyau lorsque le paquet est accepté (lors de la lecture sur une carte réseau, ou bien lors de la génération d'un paquet par une application locale).

Les commandes MANGLE d'`iptables` permettent d'effectuer ce marquage :

`iptables` est un outil d'administration pour le filtrage de paquet IPv4 et NAT.
L'option `-t` permet de spécifier le paquet correspondant à la table sur laquelle la commande doit s'appliquer. La table, dans notre cas est `mangle`. Cette table est utilisée pour des altérations de paquets spécialisés.
L'option `-A` ou `--append` introduit une chaîne de spécification de règles. L'endroit où les règles seront ajoutées à la fin de la chaîne sélectionnée. Dans notre cas, ce sera donc en sortie (`OUTPUT`).
L'option `-o` ou `--out-interface` introduit l'interface vers laquelle le paquet va être envoyé.
`-p` ou `--protocol` est l'option permettant de spécifier le protocole employé, tel que `udp`.
C'est ensuite le port de destination qui est saisie à la suite de `--dport`.
L'option `-j` ou `--jump` spécifie la cible de la règle.
`--set-mark` précise un marquage interne au noyau Linux du paquet. C'est pourquoi il est besoin d'y avoir pour chaque port une ligne de commande `iptables`.
La dernière commande iptables sert à afficher toutes les règles de la table `mangle`.

Par défaut il n'est pas initialisé. Il est possible de l'assigner arbitrairement avec des options d'`iptables`. Par exemple, pour assigner des marquages à nos flux préalablement constitués, on va modifier les lignes 9, 10 et 11 :

```
iptables -t mangle -F

iptables -t mangle -A OUTPUT -o eth1 -p udp --dport 5000 -j MARK –set-mark 1
iptables -t mangle -A OUTPUT -o eth1 -p udp --dport 5001 -j MARK –set-mark 2
iptables -t mangle -A OUTPUT -o eth1 -p udp --dport 5002 -j MARK –set-mark 3

tc filter add dev eth1parent 10: protocol ip handle 1 fw flowid 10:10 (9)
tc filter add dev eth1parent 10: protocol ip handle 2 fw flowid 10:20 (10)
tc filter add dev eth1parent 10: protocol ip handle 3 fw flowid 10:30 (11)
```

Pour afficher les opérations effectuées par `iptables`, nous allons placer la commande :

```
iptables -t mangle -L
```

On note qu'il était également possible d'effectuer un marquage "physique" (directement sur le paquet IP) en utilisant leurs champs TOS grâce aux commandes suivantes :

```
# marquage TOS
iptables -t mangle -A OUTPUT -o eth0 -p UDP --dport 5000 -j TOS –set-tos 0x10
iptables -t mangle -A OUTPUT -o eth0 -p UDP --dport 5001 -j TOS –set-tos 8
iptables -t mangle -A OUTPUT -o eth0 -p UDP --dport 5002 -j TOS –set-tos 4
```

Pour le champ TOS, toutes les valeurs ne sont pas possibles, c'est un code sur un caractère construit à la manière des codes du CHMOD du système de fichier UNIX. Ce manque de souplesse nous a fait préférer le marquage avec MARK.

Le script complet pour les expérimentations est donc le suivant (en tenant compte du marquage) :

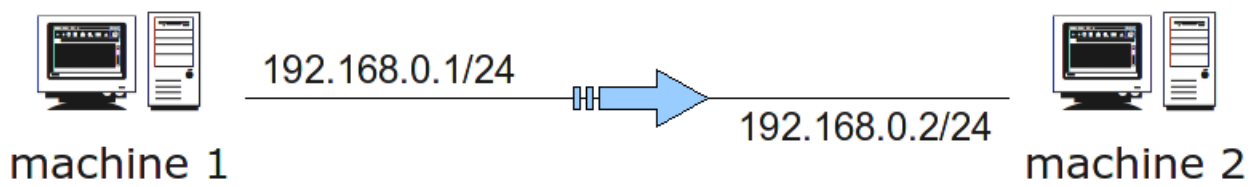
```
#!/bin/sh
# Définition des classes de Qualité de Service

# tc qdisc replace dev eth1 root handle 10: htb
tc qdisc del dev eth1 root
tc qdisc add dev eth1 root handle 10: htb
tc class add dev eth1 parent 10: classid 10:1 htb rate 1048576
tc class add dev eth1 parent 10:1 classid 10:10 htb rate 10240 ceil 10240
tc class add dev eth1 parent 10:1 classid 10:20 htb rate 204800 ceil 1048576
tc class add dev eth1 parent 10:1 classid 10:30 htb rate 716800 ceil 1048576

# Activation du partage équitable pour chaque file
tc qdisc add dev eth1 parent 10:10 handle 40: sfq
tc qdisc add dev eth1 parent 10:20 handle 50: sfq
tc qdisc add dev eth1 parent 10:30 handle 60: sfq

# Marquage des flux
iptables -t mangle -F
iptables -t mangle -A OUTPUT -o eth1 -p udp --dport 5000 -j MARK –setmark 1
iptables -t mangle -A OUTPUT -o eth1 -p udp --dport 5001 -j MARK –setmark 2
iptables -t mangle -A OUTPUT -o eth1 -p udp --dport 5002 -j MARK –setmark 3

tc filter add dev eth1 parent 10: protocol ip handle 1 fw flowid 10:10
tc filter add dev eth1 parent 10: protocol ip handle 2 fw flowid 10:20
tc filter add dev eth1 parent 10: protocol ip handle 3 fw flowid 10:30

# Affichage des opérations via tc
tc -s -d class show dev eth0

# Affichage des opérations via iptables
iptables -t mangle -L
```

Tentative de saturation des files d'attente

Afin de tester les règles mises en place précédemment, nous avons créé un script MGEN qui vise à simuler différents trafics. Nous allons, pour simuler un débit de 1 Mbps, envoyer 512 paquets de 256 octets chaque seconde durant la simulation (256 octets font 2048 bits, ce qui est un multiple de la taille des files d'attentes que nous avons utilisées).

Pour celà, on va générer du trafic entre deux machines reliées directement l'une à l'autre, selon la configuration suivante :

- Machine A : 192.168.0.1/24
- Machine B : 192.168.0.2/24



On configure chaque interface correspondante avec la commande suivante :

```
# ifconfig eth1 192.168.0.X netmask 255.255.255.0
```

Sur la première machine, trois scripts MGEN d'émission de trafic seront lancés dans trois processus différents comme suit :

```
# mgen input source.mgn
```

La génération de flux sous MGEN s'utilise avec la syntaxe suivante dans les fichiers .mgn :

```
0.0 ON 1 UDP DST 192.168.0.2/5000 PERIODIC [512.0 256]
30.0 OFF 1
```

Cette ligne permet de générer :

- **0.0** seconde après le démarrage (immédiatement), un flux nommé **1** en **UDP** avec comme IP de destination **192.168.0.2** sur le port **5000** un flux **PERIODIC** qui envoit **512** paquets de **256** octets par seconde. Ce flux s'arrête au temps **30.0**.

Le premier script, pour les communications UDP à destination du port 5000, est le suivant :

```
0.0 ON 1 UDP DST 192.168.0.2/5000 PERIODIC [512.0 256]
30.0 OFF 1
90.0 ON 1 UDP DST 192.168.0.2/5000 PERIODIC [512.0 256]
120.0 OFF 1
150.0 ON 1 UDP DST 192.168.0.2/5000 PERIODIC [512.0 256]
180.0 OFF 1
180.0 ON 1 UDP DST 192.168.0.2/5000 PERIODIC [512.0 256]
210.0 OFF 1
```

Le second script (à exécuter simultanément, dans un processus séparé) qui génère les communications UDP vers le port 5001 est le suivant :

```
30.0 ON 1 UDP DST 192.168.0.2/5001 PERIODIC [512.0 256]
60.0 OFF 1
90.0 ON 1 UDP DST 192.168.0.2/5001 PERIODIC [512.0 256]
120.0 OFF 1
120.0 ON 1 UDP DST 192.168.0.2/5001 PERIODIC [512.0 256]
150.0 OFF 1
180.0 ON 1 UDP DST 192.168.0.2/5001 PERIODIC [512.0 256]
210.0 OFF 1
```

Le dernier script (à exécuter simultanément, dans un processus séparé) qui génère les communications UDP vers le port 5002 est le suivant :

```
60.0 ON 1 UDP DST 192.168.0.2/5002 PERIODIC [512.0 256]
90.0 OFF 1
120.0 ON 1 UDP DST 192.168.0.2/5002 PERIODIC [512.0 256]
150.0 OFF 1
150.0 ON 1 UDP DST 192.168.0.2/5002 PERIODIC [512.0 256]
180.0 OFF 1
180.0 ON 1 UDP DST 192.168.0.2/5002 PERIODIC [512.0 256]
210.0 OFF 1
```

Ces trois scripts sont nommés respectivement *generate1.mgn*, *generate2.mgn* et *generate3.mgn*. On pourra les démarrer simultanément sur la machine qui envoie, à l'aide du script shell suivant :

```
#!/bin/bash
mgen ipv4 input generate1.mgn &
mgen ipv4 input generate2.mgn &
mgen ipv4 input generate3.mgn &
```

Le script pour la réception *destination.mgn* est quant à lui constitué des lignes suivantes :

```
0.0 LISTEN UDP 5000
0.0 LISTEN UDP 5001
0.0 LISTEN UDP 5002
```

Le script de réception est exécuté sur la machine qui reçoit comme suit :

```
mgen input destination.mgn [>> file]
```

Où `file` est le fichier de trace où l'on désire conserver les données.

/!\ La configuration de la QoS est effectuée uniquement sur la machine 1 qui envoie.

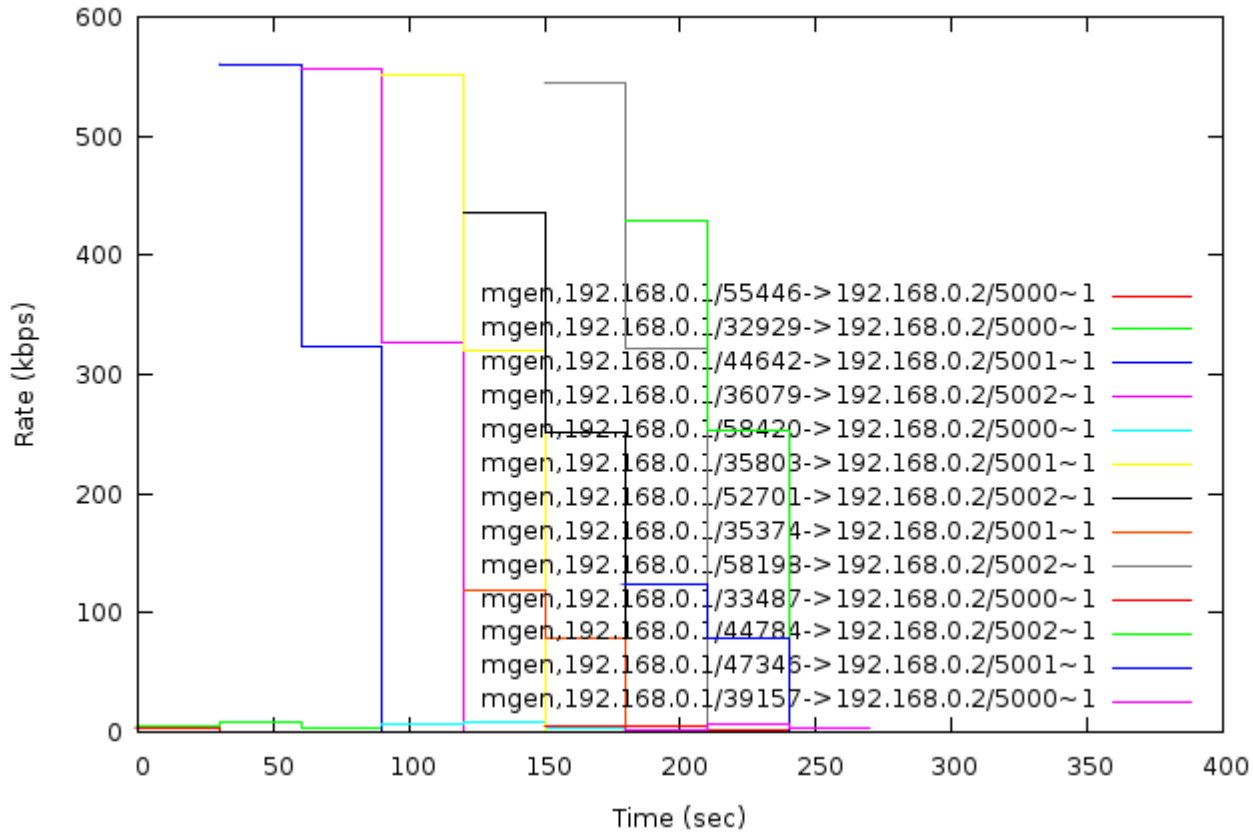
On traite le contenant les données reçues fichier grâce à l'outil TRPR, puis on affiche le graphique généré grâce aux commandes :

```
# trpr input log history 210 window 30 mgen real > data
# gnuplot -persist data
```

On spécifie à la commande TRPR que notre transmission se fait sur 210 secondes au total, par tranche de 30 secondes.
L'option `-persist` de GNUPLOT permet de garder la fenêtre affichée le temps de l'étude.

ETUDE DU RÉSULTAT

Voici le graphique qui est issu de cette manipulation :



On observe que par effet de bord, on a une poursuite de la réception des données même après la fin de l'émission.

À l'aide des données extraites et utilisées pour le traçage des courbes, on extrait les débits observés dans chacun des scénarios.

t(s)	Port 5000 (classid 10:10)		Port 5001 (classid 10:20)		Port 5002 (classid 10:30)	
	généré	**envoyé**	**généré**	**envoyé**	**généré**	**envoyé**
0	**1 Mb/s**	9,00 kb/s				
30			**1 Mb/s**	560,26 kb/s		
60					**1 Mb/s**	556,64 kb/s
90	**1 Mb/s**	6,08 kb/s	**1 Mb/s**	550,84 kb/s		
120			**1 Mb/s**	119,19 kb/s	**1 Mb/s**	435,81 kb/s
150	**1 Mb/s**	5,25 kb/s			**1 Mb/s**	544,63 kb/s
180	**1 Mb/s**	2,59 kb/s	**1 Mb/s**	124,38 kb/s	**1 Mb/s**	429,06 kb/s

Le système est constamment en saturation et ne permet à aucun moment d'avoir le trafic généré au départ sur chaque port. Lorsque le trafic est généré sur plusieurs ports on remarque que la diminution de débit en sortie de la machine est fonction des valeurs de qualité de service définies : la file ayant un plus grand débit usuel sera favorisée.

La valeur plafond (ceil) définie pour chaque file, ni celle de la file parent, n'est jamais atteinte : au mieux on peut bénéficier de 60% du débit maximum autorisé. En revanche, les trafics de classe 10:20 et 10:30 sont capables d'utiliser un *burst* supérieur à leur débit classique lorsqu'ils sont utilisés seuls, ou avec des débits faisant en sorte que la file parente délivre au plus 90% de sa capacité.

Ces mesures mettent en évidence le fonctionnement des files de discipline SFQ (*Stochastic Fair Queueing*), car on observe que SFQ se base sur un algorithme de hachage qui approxime un fonctionnement équitable en fonction du hachage obtenu. Par conséquent, il se peut que deux types de paquets soient hachés de la même façon, ce qui a pour conséquence de limiter le débit en dessous de sa valeur réelle, ici de 10% en moyenne.

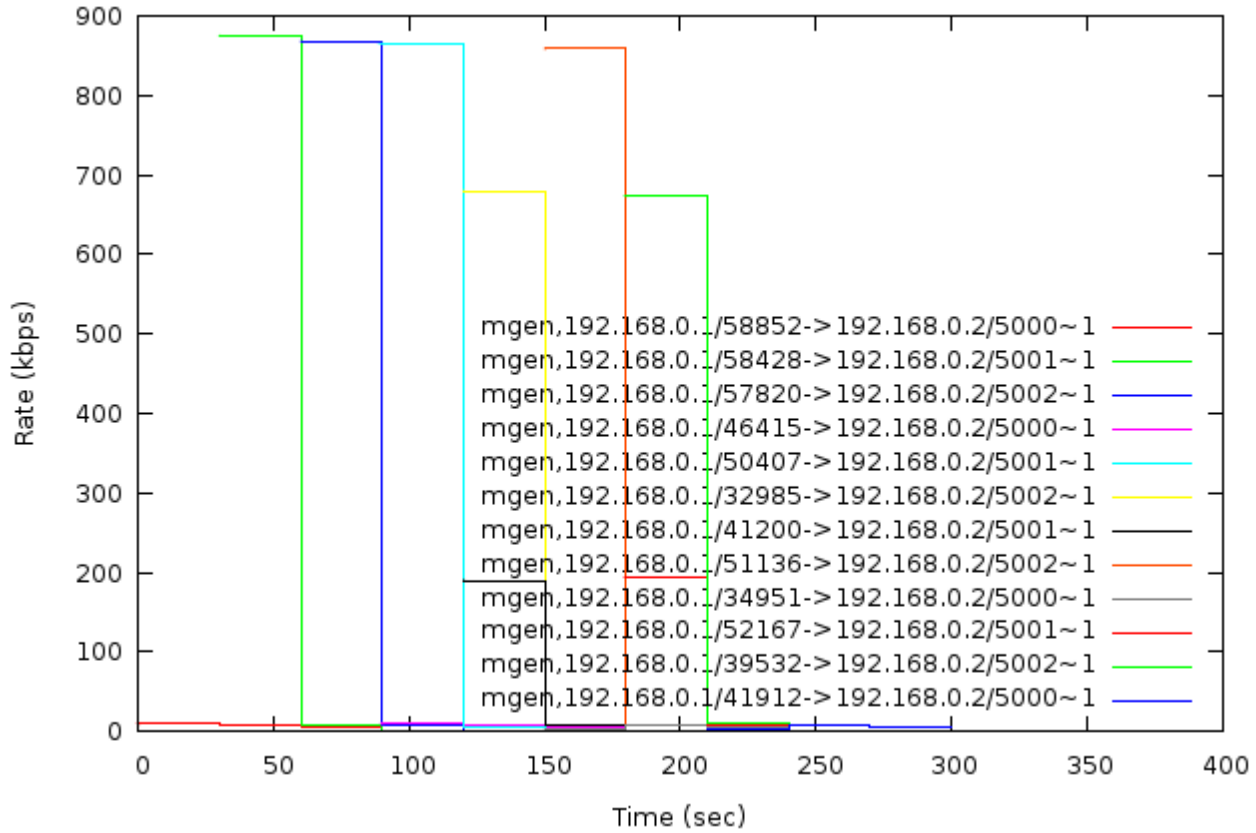
Influence de la politique de gestion des files

À présent, on va modifier la politique d'une des classes pour voir l'effet sur le trafic.
Dans le script de configuration, on change les lignes 6, 7 et 8 de la configuration initiale comme suit :

```
# Activation du partage équitable pour chaque file
tc qdisc add dev eth1 parent 10:10 handle 40: pfifo
tc qdisc add dev eth1 parent 10:20 handle 50: sfq
tc qdisc add dev eth1 parent 10:30 handle 60: sfq
```

NB : le noyau utilisé ne supporte pas la politique *pfifo_fast*, mais uniquement la politique *pfifo* (pour *Priority FIFO*).

Les scripts de tests de saturation n'ont pas besoin d'être modifiés puisque le type de file ne rentre pas en ligne de compte dans ces fichiers (ce paramètre n'est pas spécifié à cet endroit).



Sur chaque flux, une seule file d'attente est gérée. Dans ce cas, *SFQ* revient à faire du *pfifo*. Les différences que nous observons sont dues au fait qu'il s'agisse de 2 manipulations distinctes et aux aléas de gestion des paquets au niveau du système (notamment le comportement du scheduler). Toutefois, les ordres sont conservés.

t(s)	Port 5000 (classid 10:10)		Port 5001 (classid 10:20)		Port 5002 (classid 10:30)	
	généré	**envoyé**	**généré**	**envoyé**	**généré**	**envoyé**
0	**1 Mb/s**	9,35 kb/s				
30			**1 Mb/s**	875,11 kb/s		
60					**1 Mb/s**	868,22 kb/s
90	**1 Mb/s**	9,35 kb/s	**1 Mb/s**	856,21 kb/s		
120			**1 Mb/s**	189,92 kb/s	**1 Mb/s**	679,59 kb/s
150	**1 Mb/s**	1,98 kb/s			**1 Mb/s**	858,79 kb/s
180	**1 Mb/s**	1,97 kb/s	**1 Mb/s**	194,56 kb/s	**1 Mb/s**	673,04 kb/s

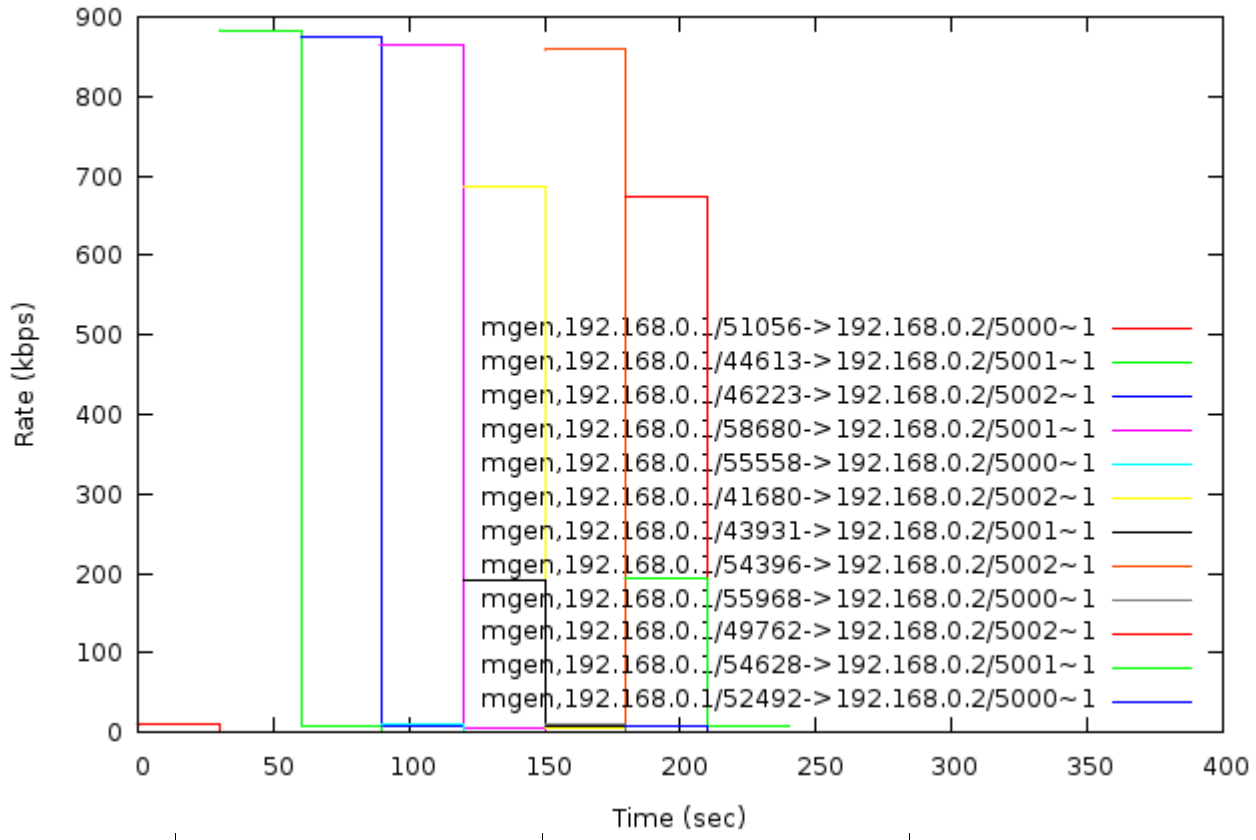
Influence de l'algorithme de gestion des files

On se propose maintenant de tester les résultats en mettant en place l'algorithme RED (pour *Random Early Detection*). Le principe est de limiter l'encombrement par anticipation du remplissage de la queue, en droppant des paquets aléatoirement quand le seuil menace d'être atteint.

Comme pour *pfifo*, on modifie le paramétrage d'une des files :

```
# Activation du partage équitable pour chaque file
tc qdisc add dev eth1 parent 10:10 handle 40: red limit 10kb min 2kb max 4kb avpkt 1000 burst 20 ecn probability 0.02
tc qdisc add dev eth1 parent 10:20 handle 50: sfq
tc qdisc add dev eth1 parent 10:30 handle 60: sfq
```

Cette première manipulation est effectuée avec RED activé pour la file 1 :



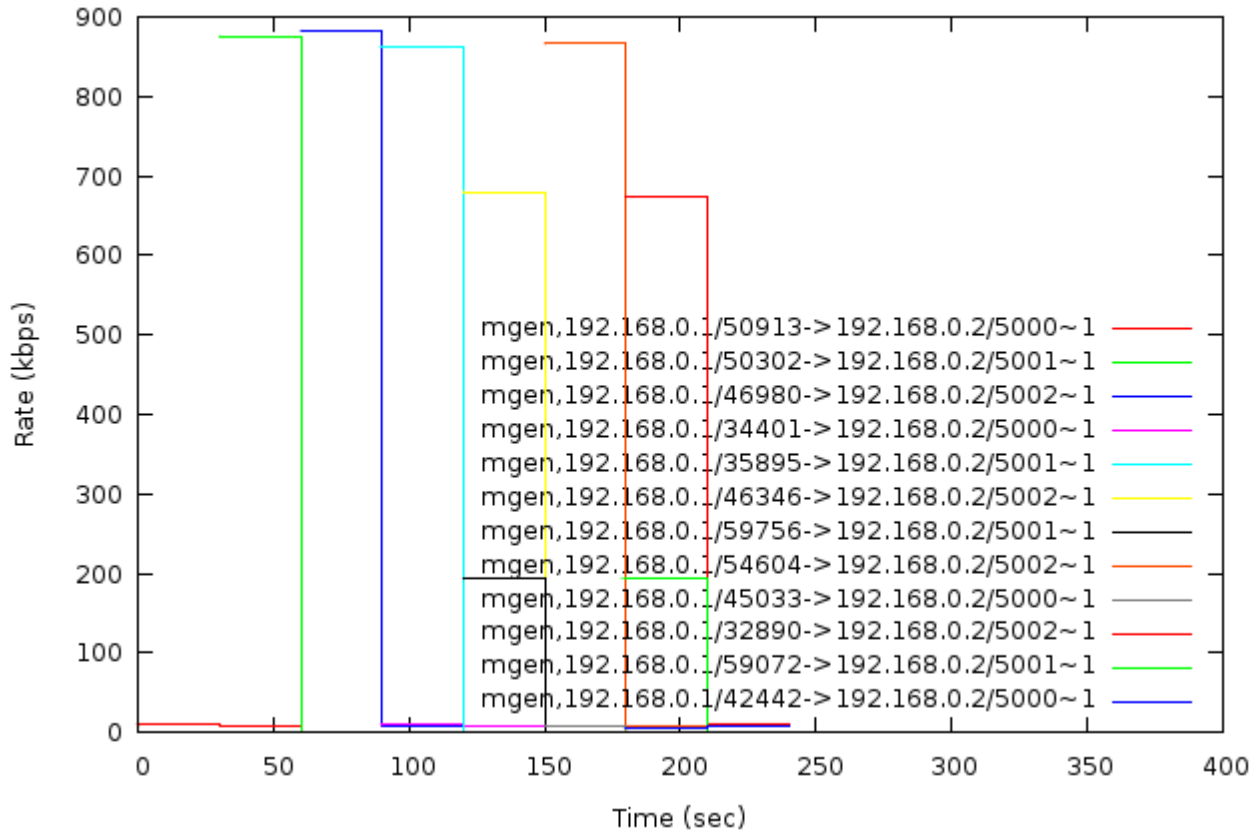
t(s)	Port 5000 (classid 10:10)		Port 5001 (classid 10:20)		Port 5002 (classid 10:30)	
	généré	**envoyé**	**généré**	**envoyé**	**généré**	**envoyé**
0	**1 Mb/s**	9,35 kb/s				
30			**1 Mb/s**	883,37 kb/s		
60					**1 Mb/s**	874,56 kb/s
90	**1 Mb/s**	9,35 kb/s	**1 Mb/s**	865,21 kb/s		
120			**1 Mb/s**	191,69 kb/s	**1 Mb/s**	686,35 kb/s
150	**1 Mb/s**	9,35 kb/s			**1 Mb/s**	858,52 kb/s
180	**1 Mb/s**	7,92 kb/s	**1 Mb/s**	194,56 kb/s	**1 Mb/s**	673,93 kb/s

On s'aperçoit que le flux transmis par la file 1 semble être beaucoup plus important, sans ralentissement pour les autres files.

Toutefois, la file 1 ayant un débit extrêmement faible, ces résultats peuvent être trompeur. Nous effectuons donc une seconde manipulation, avec cette fois la politique RED sur la file 2 :

```
# Activation du partage équitable pour chaque file
tc qdisc add dev eth1 parent 10:10 handle 40: sfq
tc qdisc add dev eth1 parent 10:20 handle 50: red limit 10kb min 2kb max 4kb avpkt 1000 burst 20 ecn probability 0.02
```

```
tc qdisc add dev eth1 parent 10:30 handle 60: sfq
```



t(s)	Port 5000 (classid 10:10)		Port 5001 (classid 10:20)		Port 5002 (classid 10:30)	
	généré	**envoyé**	**généré**	**envoyé**	**généré**	**envoyé**
0	**1 Mb/s**	9,35 kb/s				
30			**1 Mb/s**	875,38 kb/s		
60					**1 Mb/s**	882,48 kb/s
90	**1 Mb/s**	9,35 kb/s	**1 Mb/s**	862,82 kb/s		
120			**1 Mb/s**	193,60 kb/s	**1 Mb/s**	680,07 kb/s
150	**1 Mb/s**	8,53 kb/s			**1 Mb/s**	866,58 kb/s
180	**1 Mb/s**	4,30 kb/s	**1 Mb/s**	193,60 kb/s	**1 Mb/s**	673,93 kb/s

On observe alors que les résultats sont sensiblement les mêmes pour la file 2, avec une influence plus marquée par rapport à la file 1 notamment.

CONCLUSION

Ce TP nous a permis de manipuler les outils GNU/Linux de gestion de la QoS dans les transmissions réseau.

Ceci pourra par exemple se révéler utile en entreprise, si le besoin apparaît de configurer un routage réseau devant prioriser certains types d'informations par rapport à d'autres dans le cadre d'une politique de QoS (ex: réseau bancaire). En effet, à l'aide d'*iptables*, de *tc* et des modules de gestions de la QoS intégrables au noyau Linux, il est possible de réaliser presque n'importe quelle limitation ou mise en valeur imaginable.